Welcome to the first issue of this, as yet unnamed newsletter (any ideas out there?). Before launching into the informative part of this issue I thought a few words about the reasons for and aims of it would be in order.

There is no doubt that the VZ200 has enabled many people who otherwise could not afford it, to get into personal computing. On a performance per dollar basis few would complain about their VZ200. However the technical and software backup has left a bit to be desired.

The purpose of this newsletter is to overcome this backup problem. It will offer a regular, inexpensive vehicle for VZ200 users to communicate with each other. I hope it will evolve as it goes to adapt to your needs.

If you have a problem or question then write to me, if I don't have the answer perhaps another reader does. Send in your pet program, others will be interested in it and I will make every effort to include it in the next issue. Above all communicate, if you don't tell me, I won't know what you want.

To help tailor future editions I have included a questionnaire, please take the time to fill it out and return it.

THE VZ200's INVISIBLE COMMANDS

Switch on your VZ200 and enter the following;

```
10 REM 100,10
20 END
POKE 31469,183
LIST
```

the listing will be as follows;

```
10
```

now before you say "big deal" and stop reading, type

```
RUN
```

your VZ200 is now in the AUTO mode.

There are 20 commands available to you using this method, I have called them invisible because you can not list them once they have been POKE'd into RAM. The utility program listed below enables you to use these extra commands without manual POKEing. Before explaining what the program does and how it does it you need to know a bit about how your VZ200 stores a BASIC program.

Each line of BASIC in RAM (as distinct from what is displayed on the screen) starts with two bytes that contain the address of the start of the next line. The next two bytes contain the Basic line number. The fifth and subsequent bytes contain the BASIC statement. The last character of each line is zero.

By entering the following command the address of the first byte of the first line will be displayed.

```
PRINT 256*PEEK(30885)+PEEK(30884)
 31465
```

Now if you PEEK at location 31465 etc. you will see how the short program listed above is stored.

address	value	displayed as
31465	246	not displayed
66	122	" "
67	10	10(line number)
68	0	
69	147	REM
70	32	space
71	49	1
72	48	0
73	48	0
74	44	,
75	49	1
76	48	0
77	0	not displayed
31478	252	first byte of next line
79	122	
80	20	20
81	0	
82	128	END
31483	0	
84	0	
85	0	

256*122+246 gives

Note the following points,

BASIC commands are stored in a single byte
alpha numeric characters are stored as ASCII codes
the first two bytes after the last line are zero.

The utility program listed opposite can be used to study the structure of BASIC in RAM. Because it is to be used with a program written by you all spaces have been left out (to save memory) and all variables start with "Z" so that you can safely use all other letters without your program and this utility interfering with each other.

To study the way BASIC is stored type in the program and RUN as follows,

```
RUN
DO YOU WANT TO DISPLAY
A BASIC LINE ? Y
LINE NUMBER ? enter a number of your choice
```

the screen will then display the selected line as it is stored in RAM. By entering lines (or a program of your own) and looking at them with this program you will soon get a good idea of how BASIC is stored.

It is clear that each BASIC command is stored as a single byte of value greater than 127. These are called "tokens". A full list of tokens is given in table 1. The table lists the token value, the VZ200 command and the TRS 80 level 2 command if it is different from the VZ200. Why list the TRS 80 commands? you may ask. Well the reason is that the BASIC interpreter in the bottom 8K of ROM in the VZ200 is similar to the TRS 80 level 2 interpreter.

UTILITY PROGRAM

```
 10 ZA=256*PEEK(30885)+PEEK(30884):CLS
 15 PRINT"DO YOU WANT TO DISPLAY"
 16 INPUT"A BASIC LINE";ZA$
 17 IFZA$="Y"THENZZ=1ELSEZZ=0:PRINT"EDIT A NEW FUNCTION"
 20 INPUT"LINE NUMBER";ZL
 25 IF ZZ=1 THEN 40
 30 INPUT "TOKEN";ZT
 35 INPUT "MARKER";ZA$:ZM=ASC(ZA$)
 40 ZX=ZA:GOSUB 120 :Z1=PEEK(ZX)
 50 ZX=ZA+1:GOSUB 120 :Z2=PEEK(ZX)
 60 ZB=256*Z2+Z1
 70 IFZB=0THENPRINT"LINE";ZL;"IS NON EXISTANT":GOTO35
 80 ZX=ZA+2:GOSUB 120 :Z1=PEEK(ZX)
 90 ZX=ZA+3:GOSUB 120 :Z2=PEEK(ZX)
100 ZN=256*Z2+Z1:PRINT@480,ZN;
110 IFZN=ZLTHEN130ELSEZA=ZB:GOTO40
120 IFZX>32767THENZX=ZX-65536:RETURNELSERETURN
130 ZA=ZA+4:ZD=0
135 IF ZZ=1 THEN 200
140 ZX=ZA+ZD:IFZX=ZBTHEN180ELSEGOSUB120
150 IFPEEK(ZX)<>ZMTHENZD=ZD+1:GOTO140
160 POKEZX,ZT
170 GOTO 350
180 PRINT"NO MARKER IN LINE";ZL:GOTO350
200 CLS
210 PRINT"LINE";ZL;"STARTS AT"
220 PRINT"ADDRESS";ZA-4
230 PRINT ZA;">";
240 ZX=ZA+ZD:IFZX=ZBTHEN330ELSEGOSUB120
250 Z1=PEEK(ZX)
260 IFZ1<128THENPRINTCHR$(Z1);ELSEGOTO280
270 ZD=ZD+1:GOTO 240
280 IFZD=0THENPRINT@60,"".ELSEPRINT""
290 PRINTZA+ZD;">";Z1;"   TOKEN"
300 ZD=ZD+1
310 PRINTZA+ZD;">";
320 GOTO 240
330 PRINT"":PRINT"LAST CHARACTER OF LINE" ;ZL
340 PRINT"IS AT ADDRESS";ZA+ZD-1
350 INPUT"PRESS RETURN TO CONTINUE";ZA$
360 GOTO10
```

change ZL to LN

VZ200 & TRS-80 TOKENS

token	VZ200	TRS-80	token	VZ200	TRS-80	token	VZ200	TRS-80
128	END		174	-	SYSTEM	209	!	
129	FOR		175	LPRINT		210	AND	
130	RESET		176	-	DISK	211	OR	
131	SET		177	POKE		212	>	
132	CLS		178	PRINT		213	=	
133	-	DISK	179	CONT		214	<	
134	-	RANDOM	180	LIST		215	SGN	
135	NEXT		181	LLIST		216	INT	
136	DATA		182	-	DELETE	217	ABS	
137	INPUT		183	-	AUTO	218	-	FRE$
138	DIM		184	CLEAR		219	INP	
139	READ		185	CLOAD		220	-	POS
140	LET		186	CSAVE		221	SQR	
141	GOTO		187	NEW		222	RND	
142	RUN		188	TAB(		223	LOG	
143	IF		189	TO		224	EXP	
144	RESTORE		190	-	DISK	225	COS	
145	GOSUB		191	USING		226	SIN	
146	RETURN		192	-	VARPTR	227	TAN	
147	REM		193	USR		228	ATN	
148	STOP		194	-	ERL	229	PEEK	
149	ELSE		195	-	ERR	230-238	-	DISK
150	COPY	TRON	196	-	STRING$	239	-	CINT
151	COLOR	TROFF	197	-	DISK	240	-	CSNG
152	VARIFY	DEFSTR	198	POINT		241	-	CDBL
153	-	DEFINT	199	-	DISK	242	-	FIX
154	-	DEFSNG	200	-	MEM	243	LEN	
155	-	DEFDBL	201	INKEY$		244	STR$	
156	CRUN	LINE	202	THEN		245	VAL	
157	MODE	EDIT	203	NOT		246	ASC	
158	SOUND	ERROR	204	STEP		247	CHR$	
159	-	RESUME	205	+		248	LEFT$	
160	OUT		206	-		249	RIGHT$	
161	-	ON	207	*		250	MID$	
162-173	-	DISK	208	/				
174	-	SYSTEM						

TABLE 1

So similar are the two interpreters that some TRS 80 features not needed for the VZ200 can still be found. For example, starting at address 6345 is a list of two letter codes used by the TRS 80 for error messages. They can be displayed by the following program.

```
 10 CLS
 20 PRINT@33,"TRS 80 ERROR CODES"
 30 A$="bb"                          b=space
 40 FOR J=0 TO 22
 50 B$=CHR$(PEEK(6345+2*J))
 60 C$=CHR$(PEEK(6346+2*J))
 70 D$=B$+C$+A$
 80 PRINT D$;
 90 NEXT J
100 END
```

Fortunately for us there are other TRS 80 remnants in our VZ200. These include the coding to execute the extra functions mentioned above.

The next question is, why can the VZ200 execute these instructions when their token is POKE'd into memory but will not accept them from the keyboard or display them? The answer can be found by considering how the interpreter recognizes and displays a BASIC command.

In ROM starting at address 5712 is a list of BASIC commands. The first character of each word has a 1 in bit 7, this adds 128 to its value and is used to mark it as the start of a new word. When the interpreter encounters a command it searches this list to see if it is a valid command and if so a token is stored in RAM. If the word encountered is not in this list then it can not be accepted as a command and can not be listed. The following program will list the stored words one at a time.

```
 10 REM DISPLAY COMMAND LIST
 20 A=5712
 30 B=PEEK(A)
 40 IF B > 127 AND K > 0 THEN 80
 50 PRINT A;B;
 55 IF B>0 THEN PRINT CHR$( B) ELSE PRINT ""
 60 K=1 : A=A+1
 70 GOTO 30
 80 INPUT "PRESS RETURN FOR NEXT WORD" ; C$
 90 K=0 : GOTO 50
100 END
```

After running this program you will have noticed that the TRS 80 commands not available (via the keyboard) to the VZ200 user have been replaced by zeros. I may be wrong, but since all the DISK commands have been replaced by zeros it may be difficult to support a floppy disk system from the VZ200 keyboard. Perhaps one of you have some comment to make in this area?

The VZ200 commands COPY, COLOR, VERIFY, CRUN, MODE and SOUND actually replace TRS 80 commands but all those tokens in table 1 with a TRS 80 command and no VZ200 command can be used in a VZ200 program, Except the DISK commands. Some are more usefull than others but they are all of some use.

52 FOR L=1 TO 400 : NEXT L

The use of part of the utility program has already been explained. To use it to edit in the new functions is quite simple. First load the utility into your VZ200, then type in your program starting at a line above 360. Enter the program as you normally do except when you want to use an "invisible" command then type a unique character instead of the command. For example,

```
1000 PRINT VARPTR(Q$)  would be entered as
1000 PRINT &(Q$)
```

The unique character can not be one that is accepted as part of a valid BASIC command, eg *,",$ and/ . I use! or & because they are easily remembered and recognized. This character is refered to by the program as the "MARKER" and is part of the input data. You should remember that after the utility is run these functions can not be listed so it is a good idea to make the line before a REMark explaining what the next line is.

After your program is entered, but before you RUN it, RUN the utility as follows,

```
RUN
DO YOU WANT TO DISPLAY
A BASIC LINE ? N
EDIT A NEW FUNCTION
LINE NUMBER ? 1000                    )  this RUN is
TOKEN ? 192          (from table 1)   )  for the example of
MARKER ? &                            )  VARPTR shown above
```

the program then returns to the first question ready for another instruction. The program has self explanatory error messages.

When all the changes have been made a BREAK command exits from the utility. For debugging it is handy to enter a GOTO the first line of your program as line 1. That way RUN will run your program and not the utility.

The only thing left to do is to explain how these extra command are used. They are listed here in the following format,

COMMAND	Accepted format	Explanation.
RANDOM	1000 &	Randomizes the seed used to calculate the random numbers when RND is called.
DEFINT	1000 & A,B,J,L etc	Defines a list of variables as integer type.
DEFDBL	as for DEFINT	Defines a list of variables as double precision, ie 14 digit accuracy.
RESUME	1000 &	Returns from an error handling routine however it is of no use until I (or one of you) works out how to get into an error handling routine.
ON-GOTO	1000 & A GOTO 100,110,120,130 etc	"A" can be any non string variable. The next line executed is 100 if A=1, 110 if A=2 etc. If A is zero or more than the number of options listed then the next line is executed.
ON-GOSUB	as for ON-GOTO	
SYSTEM	1000 &	On the VZ200 this command will reset the Z80 in the same way as switching off the computer. The only use I have made of it is as a security device, see next month.
DELETE	1000 & 1234	Deletes a single line of BASIC from within a program.

AUTO 1 & 1000,5 Puts the VZ200 in AUTO mode. It is never used as part of a program so is best placed in line 1.

VARPTR 1000 A = &(B) or PRINT &(B) "A" is a non string variable and "B" is any variable. The function returns the address of the variable in RAM.

ERL 1000 A=& or PRINT & Returns the line at which the last error occured.

ERR 1000 A=&/2+1 or PRINT &/2+1 Returns the error number for the last error. See table 2 for a list of numbers.

MEM 1000 A=& or PRINT & Returns the amount of free memory.

FRE 1000 A=&(B$) or PRINT &(B$) Returns the amount of free string space.

POS 1000 A=&(0) or PRINT &(0) Returns cursor position

CINT CSNG and CDBL convert variables to integer, single and double precision however I have not yet worked out how to use them.

FIX Used the same as INT and does the same thing for positive numbers, however negative numbers are treated as follows, -1.52 gives -2 for INT but -1 for FIX. That is, FIX truncates a number and INT takes the next lowest whole number.

STRING$ 1000 A$=&(10,"*") or PRINT&(10,"*") This returns a string consisting of N (in this case 10) substrings (in this case *) So the above example would set A$ to or print **********.

NEXT MONTH

Next month I will talk a bit about how variables are stored, how to protect your programs from unauthorized users some ways of storing machine code programs inside your BASIC programs. I will also try to answer some of the questions received recently. Also the layout and print quality will be much improved for issue 2. Cheers and happy computing until next month.

Addendum for page 1

This is issue #1 July 1984.

John Waters
30 Seventh Ave.
CHELTENHAM
S.A. 5014

476838

ERROR NUMBERS

number returned by PRINT(ERR/2+1)

1 NEXT without FOR	9 subscript out of range	17 can not continue
2 syntax error	10 redimensioned array	18 no RESUME
3 RETURN withot GOSUB	11 division by zero	19 RESUME without
4 out of data	12 illegal direct	20 unprintable error
5 illegal function	13 type mismatch	21 missing operand
6 overflow	14 out of string space	22 bad file data
7 out of memory	15 string too long	23 disk command
8 undefined line	16 string formula too complex	

Table 2